

### REMARKS

Claims 1-6, 9-19, 22-31, 34-44, 47-50, and 53 are pending in the present application. Claims 7-8, 20-21, 32-33, 45-46, 51-52 and 54-57 have been canceled and claims 1, 3-4, 6, 9-16, 19, 22-26, 29, 31, 34-41, 44, 47-50, and 53 have been amended. Reconsideration of the claims is respectfully requested.

#### **I. Objection to Claim 45**

The examiner rejects claim 45 due to a typographical error. Claim 45 has been canceled in this response, thereby rendering the objection moot.

#### **II. 35 U.S.C. § 112, Second Paragraph: Claims 47-49 and 54**

The examiner rejects claims 47-49 and 54 as being indefinite for failing to particularly point out and distinctly claim the subject matter, which applicants regard as the invention. Claim 54 has been canceled in this response, thereby rendering the rejection moot. Claim 47 has been amended accordingly, thereby overcoming the rejection.

#### **III. 35 U.S.C. § 101: Claims 26-57**

The examiner has rejected claims 26-57 as being directed towards non-statutory subject matter. Applicants have amended these claims accordingly. Claim 26 now recites, "A computer-readable medium having stored in storage thereon computer-executable instructions..." Claim 41 now recites, "A software product for indicating when changes to values of data fields in a document have occurred, said software product stored in storage..." Claim 51 now recites, "wherein said computer program is stored in storage." Claim 53 now recites, "A computer program product stored in storage, said computer program product..."

Each of these amendments specify that the claimed subject matter is embodied in a tangible medium, and thus each of the corresponding claims is statutory subject matter. Support for the claim amendments are found on page 5, line 10 of the specification. No new matter has added by these amendments.

#### **IV. 35 U.S.C. § 102, Anticipation: Claims 1, 3-5, 7-11, 13-16, 19-22, 24-26, 28-30, 32-36, 38-41, 43, 45-47 and 49-57**

The examiner rejects claims 1, 3-5, 7-11, 13-16, 19-22, 24-26, 28-30, 32-36, 38-41, 43, 45-47 and 49-57 as being anticipated by *Getchius et al.*, Generic Object for Rapid Integration of

Data Changes, U.S. Patent No. 6,496,843 (Dec. 17, 2002) (hereinafter "*Getchius*"). This rejection is respectfully traversed.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case, each and every feature of the presently claimed invention is not identically shown in the cited reference, arranged as they are in the claims.

Amended claim 1 contains a portion of the limitations of claim 14. Claim 1 as amended is as follows:

1. A method of indicating when changes to values of data fields have occurred, said method comprising the steps of:
  - storing a first plurality of values in at least one first object, said first plurality of values comprising initial values of a plurality of data fields in a document, wherein said document comprises a notebook and wherein said notebook comprises at least one panel;
  - storing a second plurality of values in at least one second object, said second plurality of values being identical to said first plurality of values;
  - receiving modifications from a user to said plurality of data fields and storing said modifications in said at least one second object as current values;
  - comparing said first and second plurality of values to determine which data fields of the plurality of data fields have initial values different from their current values;
  - creating a plurality of records identifying all data fields of the plurality of data fields having initial values different from their current values as determined in the comparing step; and
  - transmitting said plurality of records to a server, said plurality of records for use in updating data in a data storage device, wherein the plurality of records are used to determine at least one of the following:
    - whether a panel of the at least one panel comprises a data field in the plurality of data fields having an initial value different from a current value of the data field; and
    - whether the notebook comprises a panel of the at least one panel which comprises a data field in the plurality of data fields having an initial value different from a current value of the data field.

All of the features of the amended claim 1 are not found in Getchius. In particular, Getchius does not show the italicized features of claim 1 as amended. Getchius does not show "whether a panel... comprises a data field... having an initial value different from a current value of the data field." Likewise, Getchius does not show "whether the notebook comprises a panel... which comprises a data field... having an initial value different from a current value of the data field."

Although Getchius compares data fields in different databases, this cited reference does not teach or disclose using those differing data fields to determine whether a panel has changed or whether a notebook has changed when a data field changes. Thus, each and every feature of amended claim 1 is not taught or disclosed in Getchius.

Nevertheless, in rejecting claim 14, the examiner states that:

Getchius teaches wherein said plurality of records are used to determine at least one of the following whether a data field has an initial value different from its current value; whether a panel comprises a data field having an initial value different from its current value; whether said notebook comprises a panel which comprises a data field having an initial value different from its current value (Getchius, col.44, lines 8-53).

Office Action of May 6, 2005, p. 8.

The cited text from Getchius is as follows:

Referring now to FIG. 53, at step 1400, the computation of the data update is performed using two complete sets of data from native sources. Generally, at step 1400, the latest set of data received such as from a data provider is submitted into the database and compared against the set that is in the existing database. All of the records in the data set are loaded in the following form. For comparison purposes, in the steps that follow there is a distinct record ID followed by a string where the string is all the fields from the record concatenated together for comparison purposes in steps that follow. In this particular instance record I.D.s are unique against the set and indexed. *As a result of processing at step 1400, the delta or difference between the two data sets is produced. Each entry in this delta or difference is classified as an insert, delete, or update operation. A record is inserted into the existing database in which identifiers are in the new version of the data set but not in the existing database. All records which have identifiers in the existing database, but not in the new version, are slated for deletion from the existing database. Records in which identifiers are in both sets, but, however have associated strings that differ are considered update records having data contents in the string that is updated for the corresponding identifiers. At step 1402, the update records which include inserts and update transactions are applied to the existing database. At step 1404, certain data post processing is performed as will be described further in the paragraphs that follow.*

FIGS. 46-54 generally describe data integration of the native source updates which are applied to the database of business listings and categories. In summary, for both business listings and categories, comparisons are made between records of the native source unfiltered database and native source update.

Referring now to FIG. 54, shown are more detailed steps of one embodiment of step 1400 involving the computation of the data update as pertaining to the native source business listings previously described. *At step 1406 a comparison is made between the existing database copy with the updated database copy by comparing the record identifiers and the string concatenation which represents the remainder of the records. At step 1410 each update record is classified as one of a matching entry, an insertion, a deletion, or an update with respect to the existing database. At step 1416, a record is determined to be matching if the record identifier and string field in the existing and updated data base copies match.*

Getchius, col. 44, ll. 8-53 (emphasis supplied).

The cited portion of Getchius describes comparing database copies by comparing record identifiers and the string concatenation which represents the remainder of the records. In addition, a difference between the data sets is produced. However, this portion of Getchius cited by the examiner does not show whether a panel “comprises a data field... having an initial value different from a current value of the data field,” as claimed. Likewise, this portion of Getchius does not show “whether the notebook comprises a panel of the at least one panel which comprises a data field in the plurality of data fields having an initial value different from a current value of the data field,” as claimed. This portion of Getchius only describes accommodating changes between database records, and does not describe whether panels or notebooks themselves have changed as a result of a change in a data field. Moreover, Getchius does not show or suggest the claimed features anywhere. Thus, Getchius does not anticipate claim 1 as amended.

Regarding claim 15, this claim has been amended into independent format, though without the new features included in claim 1 as amended. Getchius does not anticipate claim 15 because Getchius does not show that the “plurality of records are used to determine whether a data field having an initial value different from its current value exists at a specified level of said hierarchy of documents,” as claimed. The examiner asserts otherwise, citing various portions of Getchius. Each of these citations are addressed in turn.

The examiner first cites to figure 7 of Getchius, which is as follows:

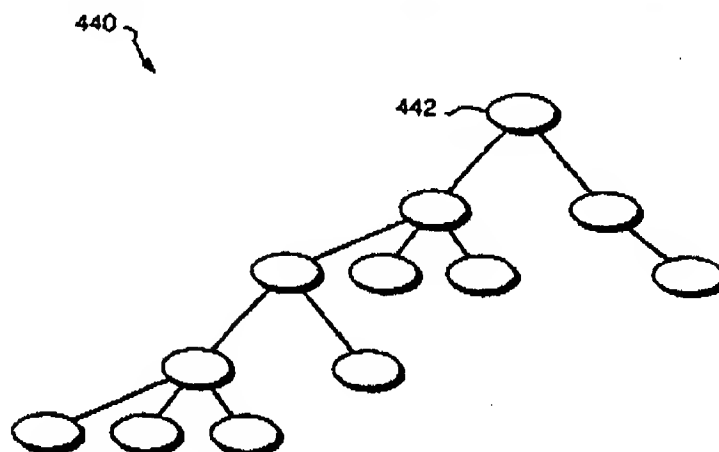


FIG. 7

The entire text describing figure 7 is as follows:

Referring to FIG. 7, a diagram illustrates a portion 440 of the PHTML execution trees 846. The portion 440 is constructed using the scripts in the PHTML files 844 and consists of a plurality of nodes corresponding to the decision points set forth in the PHTML scripts and a plurality of C++ objects and HTML pages that are executed and/or passed to the browser in response to reaching a node corresponding thereto. Thus, for example, a node 442 can correspond to a PHTML if-then-else statement having two possible outcomes wherein one branch from the node 442 corresponds to one outcome (i.e., the conditional statement evaluates to true) and another branch from the node 442 corresponds to another outcome (i.e., the conditional statement evaluates to false). Such a structure may be implemented in a conventional manner given a scripting language such as that described above in connection with the PHTML language. That is, implementing such a tree structure using a scripting language is straightforward to one of ordinary skill in the art using conventional techniques in a straightforward manner.

Representing the documents (business listings) of the databases 812, 814 as generic objects facilitates modifying the documents, or a subset thereof, without modifying the parser 866. For example, if an attribute is added to some of the objects, then it is only necessary to modify the objects (schema and data) that will contain that attribute and to also modify the PHTML files 844 to include new scripting to handle that new attribute. The scripting may include statements to determine if the particular attribute exists for each object. For example, suppose the business listings were in black and white and then color was added to some of the listings. The color attribute could be added to some, but not all, of the objects only in normalized form. Once the new color attribute has been added, the denormalized versions of all of the objects would contain a data space for the attribute, but the objects that do not possess a color attribute will have

a null marker. The PHTML files 844 can be modified to test if the color attribute is available in a particular object (e.g., to test for a null value) and to perform particular operations (such as displaying the color) if the attribute exists or, if the attribute does not exist for a particular object, displaying the object in black and white. In this way, the color attribute is added to some of the objects without modifying the parser 866 and without modifying existing objects that do not contain the attribute.

For each query that is presented to the query engine 862, the query engine 862 determines whether the query is found in the data query cache 850 or whether it is necessary to perform a query operation using the Verity software (discussed elsewhere herein) and the term list 836. In either instance, the results of the query are provided by the query engine 862 to the generic object dictionary 860 in a form set forth above in connection with the description of FIG. 6. *The parse driver 858 and PHTML execution trees 846 then operate on the generic object dictionary 860 to determine what data is displayed to the user by the browser 824.* In some instances, the PHTML execution trees 46 may require the parse driver 858 to obtain additional data from the databases 812, 814 through the data manager 864. For example, in instances where the categories corresponding to the retrieved documents (business listings) are displayed, the PHTML execution trees 846 may cause the parse driver 858 to obtain information from the generic object dictionary 860 that identifies each category and the number of listing corresponding to each category. *Then, the portion of the PHTML execution trees 846 may cause the parse driver 858 to use the data manager 864 to access additional information from the databases 812, 814, such as the names of the categories corresponding to the category identifiers provided in the generic object dictionary 860.*

Getchius, col. 20, l. 52 through col. 21, l. 53 (emphasis supplied).

This part of Getchius describes a portion of the PHTML execution trees, the construction of the trees, and the use of the execution trees. As shown by the emphasized portions of text, the execution trees “operate on the generic object dictionary 860 to determine what data is displayed to the user by the browser 824” and “cause the parse driver 858 to use the data manager 864 to access additional information from the databases.” None of the cited text and nothing in Figure 7 show that a plurality of records are used to determine whether a data field having an initial value different from a current value exists at a specified level of said hierarchy of documents, as claimed. Thus, the examiner’s assertions are incorrect. Accordingly, Getchius does not show each and every feature of claim 15.

The examiner next cites the following portion of Getchius for the proposition that Getchius shows the features of claim 15:

As generally described, the PHTML files 844 of FIG. 4 are generally HTML instructions as interpreted generally by a browser with additional embedded processing instructions. Generally, the PHTML execution tree

846 may be implemented as a C++ applet class with various execute methods which are conditionally performed based upon the evaluation of certain conditions as indicated in the PHTML scripting language statements. Each of the PHTML files 844 may be expanded and evaluated in accordance with the particular conditions of the user request. The first time a PHTML file is accessed, it is expanded and the expanded version is placed in the PHTML execution tree 846 of FIG. 4. Subsequent accesses to the same PHTML file result in the conditional evaluation of the stored and expanded PHTML file in accordance with the run time performance and evaluation of a user request, as from browser 824.

An HTML page is generally formed and displayed to the user. For example, the HTML page may be formed by the parser after interaction with the data manager and query engine to select a specific number of items to be displayed to the user. The HTML page may be stored in the page cache 848. The page cache generally includes a naming convention such as a file system in which the name of the file corresponds to the arguments and parameters of the query. The technique for forming the name is described in other paragraphs of this application.

Getchius, col. 15, l. 45 through col. 16, l. 3.

The cited text describes expandable PHTML execution trees, which are "evaluated." The evaluation is performed "in accordance with the run time performance and evaluation of a user request, as from browser 824." The PHTML trees are used to increase the efficiency of the query. Thus, the user request is for performing a database query, not for updating a database and not for tracking a specified level of a hierarchy of documents have changed data fields, as claimed. See, for example, the following portion of Getchius:

Also shown in FIG. 4 are the PHTML execution tree 846, the page cache 848, and the PHTML file store 844. *Generally, the PHTML execution tree 846 includes an expanded version of a PHTML file requested from the PHTML file 844 as the result, for example, of a user query.* PHTML generally is a modified version of the HTML language, which is a markup language according to the Standardized General Markup Language (SGML) standard, capable of interpretation by browsers, such as a Netscape browser. PHTML generally is a scripted version of HTML with conditional statements that provide for alternate inclusion of blocks of HTML code in a resulting HTML page transmitted to a browser in accordance with certain run time query conditions. *The expanded version of a PHTML file may be described as a parse tree representing parsed and expanded PHTML files.* For example, if a PHTML file conditionally includes accesses to other PHTML files or various portions of HTML commands, the parse tree structure reflects this in its representation of the parse tree which is cached in the PHTML execution tree 846. *Upon a subsequent request for the same PHTML file, the cached, expanded version is retrieved from the PHTML execution tree 846 to increase system efficiency, thereby decreasing user response time for the subsequent query.*

Page 18 of 25

Kovan et al. - 09/995,238

Getchius, col. 7, ll. 44-67.

Again, the cited text does not show "tracking a specified level of a hierarchy of documents have changed data fields," as claimed. Accordingly, Getchius does not show in the cited sections that a plurality of records are used to determine whether a data field having an initial value different from a current value exists at a specified level of said hierarchy of documents, as recited in claim 15. Likewise, Getchius is devoid of disclosure regarding this feature.

Next, the examiner cites the following portion of Getchius for the proposition that Getchius shows the features of claim 15:

Referring now to FIG. 53, at step 1400, the computation of the data update is performed using two complete sets of data from native sources. Generally, at step 1400, the latest set of data received such as from a data provider is submitted into the database and compared against the set that is in the existing database. All of the records in the data set are loaded in the following form. For comparison purposes, in the steps that follow there is a distinct record ID followed by a string where the string is all the fields from the record concatenated together for comparison purposes in steps that follow. In this particular instance record I.D.s are unique against the set and indexed. As a result of processing at step 1400, the delta or difference between the two data sets is produced. Each entry in this delta or difference is classified as an insert, delete, or update operation. A record is inserted into the existing database in which identifiers are in the new version of the data set but not in the existing database. All records which have identifiers in the existing database, but not in the new version, are slated for deletion from the existing database. Records in which identifiers are in both sets, but, however have associated strings that differ are considered update records having data contents in the string that is updated for the corresponding identifiers. At step 1402, the update records which include inserts and update transactions are applied to the existing database. At step 1404, certain data post processing is performed as will be described further in the paragraphs that follow.

FIGS. 46-54 generally describe data integration of the native source updates which are applied to the database of business listings and categories. In summary, for both business listings and categories, comparisons are made between records of the native source unfiltered database and native source update.

Referring now to FIG. 54, shown are more detailed steps of one embodiment of step 1400 involving the computation of the data update as pertaining to the native source business listings previously described. At step 1406 a comparison is made between the existing database copy with the updated database copy by comparing the record identifiers and the string concatenation which represents the remainder of the records. At step 1410 each update record is classified as one of a matching entry, an insertion, a deletion, or an update with respect to the existing database. At



step 1416, a record is determined to be matching if the record identifier and string field in the existing and updated data base copies match.

Getchius, col. 44, ll. 8-53.

The cited portion of Getchius describes comparing database copies by comparing record identifiers and the string concatenation which represents the remainder of the records. In addition, a difference between the data sets is produced. However, this portion of Getchius does not show that a plurality of records are used to determine whether a data field having an initial value different from a current value exists at a specified level of said hierarchy of documents, as claimed.

Next, the examiner cites the following portion of Getchius for the proposition that Getchius shows the features of claim 15:

Once the data modifications are incorporated into the Backoffice component, the data updates, including the updates to advertisement data and other data associated with each business listing, may be propagated to the Front End Server component. The non-text or multimedia data, for example, as included in advertisements with image files, may be transferred to the Front End Server from the Backoffice using multimedia transfer techniques, as generally described in other sections of this description. The updates to the Primary Database included in the Front End Server may be communicated as a table of commands created in the Backoffice component and transferred, as by a network connection, to the Front End Server. Generally, in this embodiment, the table created in the Backoffice includes an application developed command language corresponding to the various types of record updates and modifications that may be included in this particular embodiment. Each of these commands may be further translated in the Front End Server into one or more actual database commands that perform the table operation. For example, an entry in the table of database update commands may be specified as follows:

COMMAND	RECORD #	OPTIONAL DATA
DELETE	1-5	

In this above example table, three fields of data may be included. A Command field specifies the type of data command. The Record #field identifies which records in the Primary Database this command applies. The Optional Data includes data that may be related to the specified command. For example, if the command were update, the data field may specify the data which is to be included in the records specified. In the above example, the command is to delete records 1-5. This single table command may be translated, for example, by software included in the Primary Database, into 5 database commands in accordance with the particular database software. The software which builds the table in the Backoffice and translates the commands into one or more database commands may be developed using a commercially available software

system that is capable of communicating with the underlying database to perform the required operations.

It should be noted also that the entire table may be transferred from the Backoffice to the Front End Server, or it may be divided into sections and updates performed for each section. Additionally, each command may be sent as a separate message in other embodiments in accordance with the number of updates and other associated computer resources and costs for each data transaction. This may vary with implementation.

Getchius, col. 56, l. 19 through col. 57, l. 5.

The cited portion of Getchius describes updating a database and including a series of commands in a table of differences. The table of commands may be transferred from the back office to the front end server. However, the cited portion of Getchius does not show that a plurality of records are used to determine whether a data field having an initial value different from a current value exists at a specified level of said hierarchy of documents, as claimed.

As shown above, Getchius does not show all of the features of claim 14 as amended. Accordingly, Getchius does not anticipate claim 15.

Independent claims 16, 26, and 41 have been amended to contain features similar to those presented in claim 1 as amended. Therefore, Getchius also does not show all of the features of the these claims.

Because claims 2-6, 9-14, 17-19, 22-25, 27-31, 34-40, 42-44, 47-50, and 53 all depend from claim 1, 16, 26, or 41, the same distinctions between *Getchius* and the invention in claim 1 can also be made for these claims. Therefore, the rejection of claims 1, 3-5, 7-11, 13-16, 19-22, 24-26, 28-30, 32-36, 38-41, 43, 45-47 and 49-57 under 35 U.S.C. § 102(e) have been overcome.

Additionally, claims 2-6, 9-14, 17-19, 22-25, 27-31, 34-40, 42-44, 47-50, and 53 claim other additional combinations of features not suggested by the reference. For example, Getchius does not show wherein steps (d), (e), or (f) are repeated for each panel in a notebook, as claimed in claim 13. Consequently, it is respectfully urged that the anticipation rejection of has been overcome.

Furthermore, *Getchius* does not teach, suggest, or give any incentive to make the needed changes to reach the presently claimed invention. Absent the examiner pointing out some teaching or incentive to implement Getchius and the claimed inventions, one of ordinary skill in the art would not be led to modify *Getchius* to reach the present invention when the reference is examined as a whole. Absent some teaching, suggestion, or incentive to modify *Getchius* in this manner, the presently claimed invention can be reached only through an improper use of hindsight using Applicants' disclosure as a template to make the necessary changes to reach the claimed invention.

**V. 35 U.S.C. § 103, Obviousness: Claims 6, 17, 27 and 42**

The examiner rejects claims 6, 17, 27 and 42 as being obvious over *Getchius*. This rejection is respectfully traversed.

The examiner states on pages 10-11 of the Office Action dated May 6, 2005 that:

Regarding dependent claim 2, which is dependent on claim 1, *Getchius* teaches wherein said document is an PHTML document containing an HTML form (*Getchius*, col.15, line 45 - col.16, line 3).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified *Getchius*' PHTML document to include an HTML document, since PHTML files are HTML instructions with embedded processing instructions.

Claim 17 is for a system of presenting the method of claim 2 and is similarly rejected under the same rationale.

Claim 27 is for a computer readable medium presenting the method of claim 2 and is similarly rejected under the same rationale.

Claim 42 is for a software product presenting the method of claim 2 and is similarly rejected under the same rationale.

Office Action dated May 6, 2005, pages 10-11.

If the Patent Office does not produce a *prima facie* case of unpatentability, then without more the applicant is entitled to grant of a patent. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Grahiak*, 769 F.2d 729, 733, 226 U.S.P.Q. 870, 873 (Fed. Cir. 1985). A *prima facie* case of obviousness is established when the teachings of the prior art itself suggest the claimed subject matter to a person of ordinary skill in the art. *In re Bell*, 991 F.2d 781, 783, 26 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1993). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). A proper *prima facie* case of obviousness cannot be established by combining the teachings of the prior art absent some teaching, incentive, or suggestion supporting the combination. *In re Napier*, 55 F.3d 610, 613, 34 U.S.P.Q.2d 1782, 1784 (Fed. Cir. 1995); *In re Bond*, 910 F.2d 831, 834, 15 U.S.P.Q.2d 1566, 1568 (Fed. Cir. 1990).

Therefore, the rejection of claims 6, 17, 27 and 42 under 35 U.S.C. § 103(a) has been overcome.

**VI. 35 U.S.C. § 103, Obviousness: Claims 6, 12, 18, 23, 31, 37, 44 and 48**

The examiner rejects claims 6, 12, 18, 23, 31, 37, 44 and 48 as obvious over *Getchius* in view of *Baillargeon et al.*, Multi-Nodal Meeting Planning System and Method, U.S. Patent Publication No. 2002/0046076 (Apr. 18, 2002) (hereinafter "*Baillargeon*"). This rejection is respectfully traversed.

The examiner states that:

Regarding dependent claim 6, which is dependent on claim 5, Getchius does not explicitly disclose a Boolean value assigned to each of said data field identifiers to indicate whether said data field has an initial value different from its current value as determined in step comparison.

Baillargeon teaches a Boolean value assigned to each of said data field identifiers to indicate whether said data field has updated (Baillargeon, page 4, paragraphs 50-52, fig.2).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Baillargeon's using Boolean into Getchius to mark data field identifiers that need to be updated, since this would have facilitate the updating of values of the data fields that need to be changed and keep track what data is updated or need to be update.

Office Action dated May 6, 2005, p. 11 (emphasis in original).

If the Patent Office does not produce a *prima facie* case of unpatentability, then without more the applicant is entitled to grant of a patent. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Grabiak*, 769 F.2d 729, 733, 226 U.S.P.Q. 870, 873 (Fed. Cir. 1985). A *prima facie* case of obviousness is established when the teachings of the prior art itself suggest the claimed subject matter to a person of ordinary skill in the art. *In re Bell*, 991 F.2d 781, 783, 26 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1993). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). A proper *prima facie* case of obviousness cannot be established by combining the teachings of the prior art absent some teaching, incentive, or suggestion supporting the combination. *In re Napier*, 55 F.3d 610, 613, 34 U.S.P.Q.2d 1782, 1784 (Fed. Cir. 1995); *In re Bond*, 910 F.2d 831, 834, 15 U.S.P.Q.2d 1566, 1568 (Fed. Cir. 1990).

The independent claims from which claims 6, 12, 18, 23, 31, 37, 44, and 48 depend have been amended. Getchius neither shows nor suggests any of the features of the independent claims as amended. Baillargeon fails to cure the lack of disclosure in Getchius in this regard. Thus, these claims are not obvious in view of Getchius at least by virtue of their dependency on the corresponding independent claims.

Additionally, in viewing the references as a whole, one of ordinary skill would look to the problems addressed by the references in determining whether to combine the references. The claims are not obvious in view of Getchius and Baillargeon when the references are viewed as a whole because the two references address different problems. Getchius is directed toward solving the problem of updating on-line database systems, as shown below:

In an on-line database query system, data from the databases is presented in a user-readable form to a user screen. Presentation software of the database query system interprets the data and determines how the information is shown to the user. When the format of the data is changed, it is often necessary to also modify the presentation software to account for the change. For example, if the data of the database represents a set of business listings that are presented to the user in black and white (i.e., without any color component), and a color feature is subsequently added to the listings, then the data in the database is modified to include the new feature and, at the same time, the presentation software is modified to include code to properly interpret and display the new feature.

However, for on-line query systems that run continuously and handle large amounts of data, it may be impractical (to both the users and the operators) to require modification of the presentation software every time there is a modification to the data format such as adding a new data attribute. This difficulty may be compounded when the data represents advertisements for businesses that may be modified/enhanced in any number ways by the represented businesses in order to facilitate the advertising effect.

Getchius, col. 1, ll. 34- 56.

On the other hand, Baillargeon is directed toward the problem of increasing functionality and security for on-line meetings, as shown below:

Although there are a variety of on-line travel and planning services available on the Internet, they are typically of limited scope and completely on-line based, i.e., an Application Service Provider ("ASP"). ASPs, such as GetThere.com, of Menlo Park, Calif., and Event411.com, of Marina del Rey, Calif., do not provide any functionality unless an Internet connections present and address only a single aspect of the operations needed to successfully plan and operate a group meeting. ASPs do not address the need of meeting planners to have complete access and control of all aspects of the group meeting. Most of ASPs have a policy that nay data entered into their systems becomes the property of the ASP service provider. There is often sensitive information involved which corporations are reluctant to release to a third party for use in the operation of a group meeting.

Baillargeon, paragraph 12.

The problems addressed by Baillargeon and Getchius are completely distinct from each other. Thus, one of ordinary skill would have no reason or motivation to look to Baillargeon for problems found in Getchius. Accordingly, the claims are not obvious in view of the references when the references are considered as a whole. Therefore, the rejection of claims 6, 12, 18, 23, 31, 37, 44 and 48 under 35 U.S.C. § 103(a) has been overcome.

**VII. Conclusion**

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance.

The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: August 8, 2005

Respectfully submitted,



---

Theodore D. Fay, III  
Reg. No. 48,504  
Yee & Associates, P.C.  
P.O. Box 802333  
Dallas, TX 75380  
(972) 385-8777  
Attorney for Applicants